Multi-Objective Packing of 3D Objects into Arbitrary Containers

Hermann Meißenhelter, René Weller and Gabriel Zachmann

University of Bremen, Germany

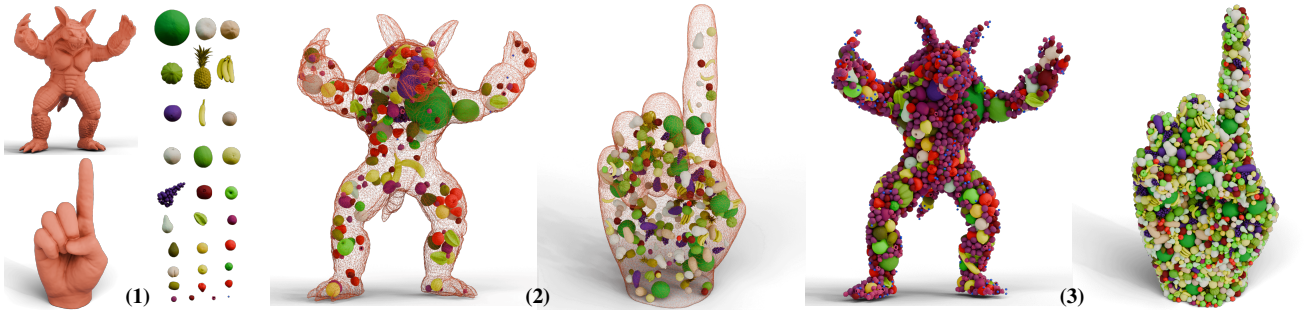

Figure 1: *(1) Example containers (Armadillo and Hand) and objects (fruits) to be packed inside, both with high polygon count as well as arbitrary shapes. (2) We can pack the fruit densely into a container, such that a given fruit distribution is achieved, yet clustering is avoided. Armadillo distribution with more smaller fruit sizes, and Hand with more larger ones. (3) Final packings: Armadillo (54% volume covered in 33 min), Hand (54.1% volume covered in 23 min).*

Abstract

Packing problems arise in numerous real-world applications and often take diverse forms. We focus on the relatively underexplored task of packing a set of arbitrary 3D objects—drawn from a predefined distribution—into a single arbitrary 3D container. We simultaneously optimize two potentially conflicting objectives: maximizing the packed volume and maintaining sufficient spacing among objects of the same type to prevent clustering. We present an algorithm to compute solutions to this challenging problem heuristically. Our approach is a flexible two-tier pipeline that computes and refines an initial arrangement. Our results confirm that this approach achieves dense packings across various objects and container shapes.

CCS Concepts

• ***Theory of computation*** → ***Packing and covering problems;*** *Computational geometry;*

1. Introduction

Packing problems have long been studied in computational geometry and operations research, dating back to Kepler's 1611 conjecture on sphere packing. Packing problems are still an active field of research, and there exist a lot of open research questions, even for very simple geometries, such as the sausage catastrophe [TGW89], not to speak about the packing of arbitrary objects.

Today, the packing of objects with arbitrary shapes, especially in 3D, is of high interest in many practically relevant scenarios. For instance, it is important in logistics to pack as many objects as possible inside a container or reduce the number of containers to minimize the shipping costs. In diamond-cutting, it is essential to reduce the wasted material , and in decorative paving, a closer placement of differently sized stones increases the surface coverage. A relatively new challenge arises in additive manufacturing: the process is very time-consuming, so it is more efficient to process many objects simultaneously and, therefore, pack them together as close as possible in a container [LtDF12].

While several researchers have addressed the problem of packing arbitrarily shaped 3D objects into arbitrarily shaped containers (e.g., [MCHW18, CRCM23]), these methods often focus purely on maximizing packing density. In contrast, our approach targets additional objectives that arise in more specialized settings, such as ensuring a predefined distribution of object types and avoiding visually unappealing clustering. These considerations emerged from a practical collaboration with an artist seeking aesthetically pleasing arrangements for 3D-printed art pieces. Similar constraints arise in industrial contexts such as cement mixture creation. New mixtures

DOI: 10.2312/egs.20251051

of non-uniformly sized particles are often tested through simulation to ensure a uniform spatial distribution with high packing density [Aa08].

Because we are not aware of an analytical solution to this multi-objective setting, we present a two-tier heuristic pipeline based on computer graphics methods—particularly distance computations and collision detection. In the first phase, we compute a feasible initial packing; in the second phase, we refine it to achieve higher density. Our examples demonstrate that this approach yields dense packings and meets user-defined distribution goals in a reasonable time. The source code is publicly available at: cgvr.cs.uni-bremen.de/research/packing

2. Related Work

Wäscher's typology [WHS07] classifies numerous packing problems; our scenario, involving many "weakly heterogeneous" items in one large 3D container, falls under the *3D irregular non-orthogonal Single Large Object Placement Problem (SLOPP)*. This class has received relatively little attention, and no analytic solution is known, so heuristic methods are typically employed to achieve high-density packings. One such heuristic was presented in additive manufacturing to optimize the layout of the objects [LtDF12]. Objects are placed randomly, and then vibration is simulated to increase the density. This leads to voids that are filled by smaller objects (*Brazil Nut Effect*).

Many algorithms for packing problems can handle only a few simple, regular shapes such as spheres or boxes. Including other or arbitrary shapes often fails because these objects are difficult to treat mathematically. For these reasons, [JW01] uses a voxelization (i.e., discretization) of complex objects in 3D. This introduces the problem of choosing the right discretization resolution for a given application because that will determine accuracy and runtime. The authors primarily employ a physically-based approach to distribute the objects in the container using several special movements, such as a *rebounding probability*, which allows objects to move diffusely [GGJW04]. A disadvantage of this approach is the high memory consumption.

Ma et al. [MCHW18] combined a continuous, local optimization with a combinatorial optimization. Each object is placed centrally into cells, and then scaling, rotation, and position are optimized iteratively until a stopping criterion is met. However, this approach does not achieve a uniform spatial distribution of the objects and is computationally very expensive.

Recently, Cui et al. [CRCM23] was able to demonstrate state-of-the-art performance compared to [MCHW18, RBSP18]. This was made possible by voxelization of the container and objects and using the Fast Fourier Transform (FFT) for collision detection. They limited the degree of rotations, which was remedied by simulating container-shaking [ZCH*24]. However, it remains unclear how efficient the FFT approach is for arbitrary container shapes compared to rectangular ones, as [CRCM23] provided computation time for only a single example, and rotations were disabled in that case. In contrast, our work primarily considers arbitrary container shapes and allows for continuous rotations, addressing these limitations directly.

Algorithm 1 Initialization: Uniformly Place and Grow

```
1: sample points P inside container
2: place objects O at P with random rotation
3: while not all object in O grown do
4:     search object type T greedily w.r.t. volume and distribution
5:     let o be the first ungrown object of type T
6:     for s ← 0.05 to 1.0 with step 1/30 do
7:         scale o to s
8:         try to resolve overlap                    ▷ up to i iterations
9:         if overlap not resolved then
10:            remove o from O
```

3. Problem Statement

Given a container $C \subset \mathbb{R}^3$, and given $T = \{O_1, \ldots, O_n\}$ a collection of object types, where each O_i is a given solid shape in $\mathbb{R}^3$. A *placement* $P = (X,t)$ is a rigid motion X (translation + rotation) and a pointer $t \in \{1, \ldots, n\}$ to a specific type of object. Find a set of placements $P_i, \ldots, P_N$ so that:

- $X_i(O_{t_i})$'s do not overlap and are contained in C
- Maximize the density $\sum_{i=1}^{N} \mathrm{vol}(X_i(O_{t_i}))/\mathrm{vol}(C)$
- Adhere to user-specific frequency in type
- Avoid spatial clustering among objects of the same type

In our evaluation, we quantify distribution accuracy and clustering.

4. Our Approach

Here, we present a set of heuristic algorithms to solve the problem stated above. In principle, our two-phase pipeline works as follows:

1. *Initialization:* First, we compute a *feasible* solution, i.e., all objects are located inside the container object and not overlapping.
2. *Optimization:* In the second phase, we try to add more objects to increase the packing density.

Note that our pipeline design is very flexible in that it allows easy integration of more heuristics in the future. In the following, we provide more details of the two phases. Finally, we describe how object overlaps are solved.

Initialization Given an empty container, the basic idea is to place objects at potentially feasible positions and resolve conflicts. Beforehand, a desired, frequency distribution of object types has to be predefined. We avoid gravitational force or container shaking [LtDF12, ZCH*24], as it would lead to the mentioned *Brazil Nut Effect* and increase clustering by size segregation. With Algorithm 1, we try to achieve all three objectives (non-clustering, distribution, density) by placing and growing objects inside a container. We start by regularly sampling points within the container at a coarse resolution, aiming for a quick initial solution. Each object is placed at one of these sampled points with a randomly assigned orientation and a type drawn from the discrete distribution (e.g., a 10% chance for an apple). Initially, objects are placed in a "shrunken", non-colliding state, then grown consecutively. However, we do not grow all objects simultaneously. Instead, we select the object type with the largest volume that is currently below its desired target frequency. If available, we then choose the first non-grown object of that type; otherwise, we don't consider this type anymore and repeat the selection process.

Once an object is chosen, we scale it incrementally, starting from a small factor (0.05) and increasing in uniform steps up to 1.0. After each scaling step, we attempt to resolve any overlaps with previously placed objects, allowing up to 25-30 iterations for this process. If overlaps cannot be resolved within i iterations, we remove the object from the set of placed objects.

Optimization The initial packings often have cavities due to the finite sampling resolution and removed objects. Obviously, these voids have different shapes and sizes. With Algorithm 2, we present a simple heuristic to identify possible voids and try to fill them, which leads to more packing density. This time, we do the sampling in higher resolution to catch the voids. Since it is difficult to place objects successfully with already placed objects, we classify the sampling points. We compute a minimal distance to the nearest object from each sampling point. To improve placement success and consider type distribution, we discard cavities falling below a threshold, which depends on the object type frequency w_i. Specifically, we compute the weighted geometric mean $\bar{V} = \prod_i V_i^{w_i}$ of the volumes V_i. Let $f(x) = \sqrt[3]{3x/(4\pi)}$ (radius from sphere volume). We then set $d = f(\bar{V})$ as our distance threshold for cavities. This volume-based approach ensures elongated shapes can still be chosen to fill a cavity. However, volume scales cubically, which biases toward larger cavities, so we further refine d by averaging it with the radius $r = f(V_i)$ of the smallest object volume V_i. The resulting distance $\frac{d+r}{2}$ is then used to reject sampling points that are too close to existing objects.

For each point, we try to insert the largest type of object that can fit in the cavity. An object is valid if its volume to radius ($f(V_i)$) is smaller than the sampling point's minimal distance. The object is placed at the current point and gradually scaled up, with overlap resolutions performed at each scaling step (55-150 iterations). We move on to the next sample point if the object reaches its full size without collisions. Otherwise, we retry with a smaller object type (randomly selected) up to a maximum of two insertion attempts. If no smaller object can be placed successfully or the maximum number of attempts is reached, we discard the current point and proceed with the next one.

Algorithm 2 Optimization: Fill Cavities

```
sample points P inside container with minimum distance d
for each p ∈ P do
    let t be the largest object type fitting the cavity at p
    scale t and resolve overlap                          ▷ i iterations
    if reached full size then                            ▷ no overlap exists
        continue                                         ▷ move on to next point
    tries ← 0
    while tries < 2 and not (full size reached) do
        tries ← tries + 1
        select random smaller object type t
        if no smaller type t exists then
            break                                        ▷ smallest type failed, skip point
        scale t and resolve overlap                      ▷ i iterations
```

Testing and Resolving Object Overlap During the placement of objects, we allow overlaps. However, the final packing should be overlap-free. Hence, we resolve the intersections with the physically-based approach (custom rigid body simulator). We use sphere packing approximations of objects for object-object collision and penalty-force computation as [WZ09] did. Besides fast computation of penetration volume (penalty force), we also can do fast distance queries. Intersecting spheres or minimal distances are computed by traversing both sphere bounding volume hierarchies. Since the container is more difficult to approximate with a volumetric sphere packing, we use a k-DOP tree [Zac98] for object-container intersection instead. A k-DOP tree is a hierarchical spatial data structure that uses discrete oriented polytopes (k-DOP) as bounding volumes. SIMD instructions speed up our bounding volume hierarchies, which use 4-ary trees. After accumulating penalty forces on each object, we normalize them and scale them by a spring-like object stiffness (giving each object-object collision equal strength). This scaled offset is added to the velocity. For object–container collisions, we compute the average normal from the intersecting container triangles, scale it by a container stiffness, and add it to the velocity. We then integrate the velocity and damp it to zero, causing objects to move only a fixed amount and stop when no collision exists. This design simplifies the tuning of penalty stiffness for both collision types and accelerates collision detection through temporal coherence (unmoved objects remain out of further checks). There is, in general, no guarantee of reaching an overlap-free state or beforehand known how many iterations i (physics updates) are needed. Therefore, we limit the iterations to a predefined hyperparameter. Moreover, we save the previous overlap-free packing in memory before we attempt to add an object. Therefore, we can quickly recover after a failed insertion attempt.

5. Results

We benchmarked four cases from the literature [MCHW18, RBSP18] (uniform type distribution) and additionally our two high-polygonal containers and fruits with regard to total computation time and the resulting packing density. Our PC had an Intel® Core™ i7-12700K CPU with 32 GB RAM. In Fig. 2 and Tab. 2, we compare our results in a similar manner to [CRCM23, ZCH*24], which also listed computation times from the original references. Our method achieves the fastest computation time while aiming for additional objectives like distribution and non-clustering, at the cost of slightly lower packing density (see Tab. 2). The initialization took up to 19% of total time and was up to 13% below optimized packing density. We compute the difference between our target object type distribution q and the actual distribution p using the *total variation distance* (TVD), which satisfies $0 \leq \mathrm{TVD}(p,q) \leq 1$ and represents the fraction of the total probability mass that must be redistributed to make the two distributions identical. We achieve almost identical distributions for initialization, and low differences after the optimized packing, except for one case (see Tab. 1). We hypothesize that the many fine-detail parts of the Armadillo make it difficult to place medium- to large-sized fruit objects, thus leading to more distribution differences. The initialization performs very well because it mainly focuses on distribution. We evaluate the spatial distribution using the *nearest neighbor index* (NNI), which allows us to determine whether there is any clustering (NNI < 1), random placement (NNI ≈ 1), or regularities (NNI > 1). Results show that our approach avoids spatial clustering of object types on average across all use cases.

Metric	Arma.	Box	Chess	Hand	Torus	Vase
TVD_{init}	0.15	0.07	0.01	0.02	0.00	0.05
TVD_{opt}	0.53	0.18	0.03	0.34	0.00	0.09
NNI_{init}	1.43	1.17	1.15	1.25	1.20	1.41
NNI_{opt}	1.20	1.14	1.16	1.24	1.25	1.41

Table 1: *Results for object type distribution (Total Variation Distance, TVD) and spatial distribution (Nearest Neighbor Index, NNI). Initialization (*init*) and optimization (*opt*) algorithm results.*

Method	Number	Time[s]	Density[%]
Romanova et al. [RBSP18]	80	42950	53.66
Ma et al. [MCHW18]	77	2700	51.3
Cui et al. [CRCM23]	80	329	51.27
Zhuang et al. [ZCH*24]	86	277	54.47
Ours	78	139	45.5

Table 2: *Comparison of time consumption and packing density for our method vs. others in the polyhedron scenario (see Fig. 2b).*

6. Conclusion and Future Work

In this work, we addressed the challenging problem of packing arbitrary 3D objects into arbitrary containers while optimizing for packing density and a given distribution across object types. Our heuristic algorithms maintain a cluster-free spatial distribution of object types and achieve dense packings using a two-phase pipeline consisting of initialization and optimization.

Results show the reliability and adaptability of our approach, making it suitable for manufacturing and many other fields. Our work could be extended by, i.e., a new algorithm searches for unconnected objects and tries to attach them to the nearest object. Additionally, we will investigate whether performing consecutive local optimizations at incrementally increasing sampling resolutions generally outperforms a single high-resolution sampling, as the actual distances might change.

Acknowledgements This research has been (partially) supported by the German Research Foundation DFG, as part of Collaborative Research Center (Sonderforschungsbereich) 1320 “EASE - Everyday Activity Science and Engineering”. Hand and fruit models are copyright by Peter Coffin Studio.

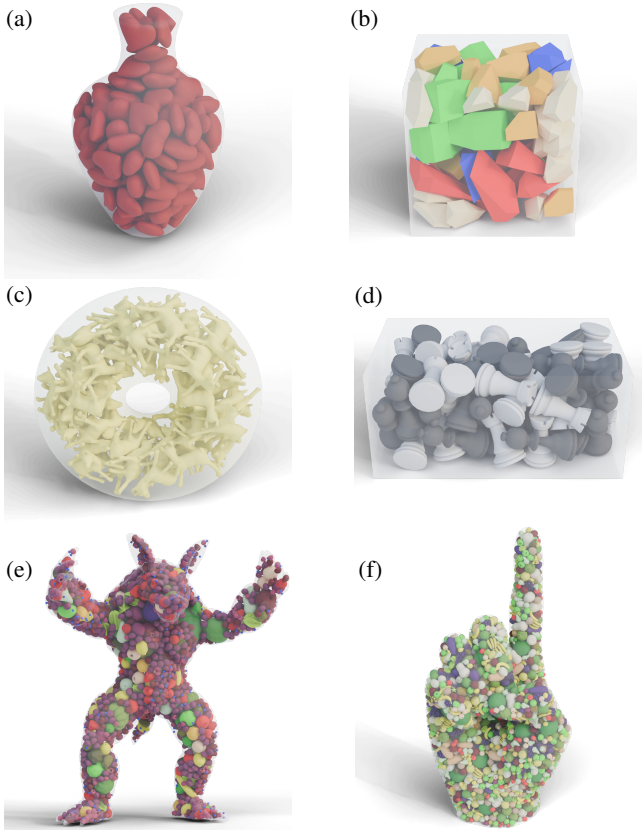

Figure 2: *Results. (a) Vase filled by 135 hearts (51.7%) in 63 s. [MCHW18]: (125 hearts, 40.3%, 1740 s). (b) Packing 78 polyhedrons (45.5%) in a box within 139 s. (c) Torus filled by 67 dogs (19.6%) in 112 s. [MCHW18]: (67 dogs, 19.6%, 3480 s). (d) Packing 62 chess pieces (32.1%) in a box within 86 s. [MCHW18]: (60 pieces, 32.5%, 2400 s). (e) Armadillo filled with 2209 fruits (54%) in 1984 s. (f) Hand packed with 2557 fruits (54.1%) in 1376 s.*

References

[Aa08] AMIRJANOV A., AND K. S.: Optimization of a computer simulation model for packing of concrete aggregates. *Particulate Science and Technology 26*, 4 (2008), 380–395. 2

[CRCM23] CUI Q., RONG V., CHEN D., MATUSIK W.: Dense, interlocking-free and scalable spectral packing of generic 3d objects. *ACM Trans. Graph. 42*, 4 (July 2023). 1, 2, 3, 4

[GGJW04] GAN M., GOPINATHAN N., JIA X., WILLIAMS R. A.: Predicting packing characteristics of particles of arbitrary shapes. *KONA Powder and Particle Journal 22* (2004), 82–93. 2

[JW01] JIA X., WILLIAMS R.: A packing algorithm for particles of arbitrary shapes. *Powder technology 120*, 3 (2001), 175–186. 2

[LtDF12] LUTTERS E., TEN DAM D., FANEKER T.: 3d nesting of complex shapes. *Procedia CIRP 3* (2012), 26 – 31. 45th CIRP Conference on Manufacturing Systems 2012. 1, 2

[MCHW18] MA Y., CHEN Z., HU W., WANG W.: Packing irregular objects in 3d space via hybrid optimization. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 49–59. 1, 2, 3, 4

[RBSP18] ROMANOVA T., BENNELL J., STOYAN Y., PANKRATOV A.: Packing of concave polyhedra with continuous rotations using nonlinear optimisation. *European Journal of Operational Research 268*, 1 (2018), 37 – 53. 2, 3, 4

[TGW89] TÓTH G. F., GRITZMANN P., WILLS J. M.: Finite sphere packing and sphere covering. *Discrete & Computational Geometry 4*, 1 (1989), 19–40. 1

[WHS07] WÄSCHER G., HAUSSNER H., SCHUMANN H.: An improved typology of cutting and packing problems. *European Journal of Operational Research 183*, 3 (2007), 1109 – 1130. 2

[WZ09] WELLER R., ZACHMANN G.: A unified approach for physically-based simulations and haptic rendering. In *Sandbox: ACM SIGGRAPH Video Game Proceedings* (Aug. 2009), ACM Press. 3

[Zac98] ZACHMANN G.: Rapid collision detection by dynamically aligned DOP-trees. In *Proc. of IEEE Virtual Reality Annual International Symposium; VRAIS '98* (Atlanta, Georgia, March 1998), pp. 90–97. 3

[ZCH*24] ZHUANG Q., CHEN Z., HE K., CAO J., WANG W.: Dynamics simulation-based packing of irregular 3d objects. *Comput. Graph. 123*, C (Nov. 2024). 2, 3, 4